Desain *S-Box* Fleksibel: Regenerasi Konstanta dan Koefisien Fungsi Linier Berbasis CSPRNG *Chaos*

Bambang Susanto[1], Alz Danny Wowor[2], Vania Beatrice Liwandouw[3]

***Abstract*—The substitution process in block ciphers usually uses static s-boxes, where the value of each s-box entry is always fixed and the functions are one-to-one. Visually, the dataset pattern in the input is also seen in the output. This will make it easier for cryptanalysts to look at patterns and can predict behavior based on the input. This study designs a flexible s-box where each entry contains a linear function, with constants and coefficients generated by CSPRNG chaos from the key input. The strength of a flexible s-box can be seen when changing key inputs (although 1 bit different) will produce different s-boxes. In addition, the same input dataset values will produce different outputs, so flexible s-boxes are one-to-many. Statistical tests show that the average correlation is in a low category. Tests on the cipher block also fulfill the Shannon Principle. The principle of Iterated Cipher with n-round is filled with $n > 5$ and $46n$ random numbers. The Avalanche Effect test on the algorithm gives better results than Twofish even though it is still below the DES.**

***Intisari*—Proses substitusi dalam blok *cipher* biasanya menggunakan *s-box* statis, dengan nilai pada setiap entri *s-box* selalu tetap, dan secara fungsi bersifat satu-ke-satu. Secara visual, pola *dataset* pada masukan terlihat juga pada keluaran. Hal ini akan mempermudah *cryptanalyst* untuk melihat pola dan dapat memprediksi perilaku berdasarkan masukan. Makalah ini merancang *s-box* fleksibel yang setiap entrinya berisi fungsi linier, dengan konstanta dan koefisien yang dibangkitkan oleh CSPRNG *chaos* dari masukan kunci. Kekuatan dari *s-box* fleksibel terlihat ketika perubahan masukan kunci (walaupun berbeda 1 bit) akan menghasilkan *s-box* yang berbeda. Selain itu, masukan nilai *dataset* yang sama akan menghasilkan keluaran yang berbeda, sehingga *s-box* fleksibel bersifat satu-ke-banyak. Uji statistik menunjukkan bahwa korelasi secara rata-rata berada pada kategori rendah. Pengujian pada blok *cipher* juga memenuhi prinsip Shannon. Prinsip *Iterated Cipher* dengan *n*-putaran dipenuhi dengan $n > 5$ dan $46n$ bilangan acak. Uji *Avalanche Effect* pada algoritme memberi hasil yang lebih baik dari Twofish walaupun masih di bawah DES.**

***Kata Kunci*—*S-box* fleksibel, blok *cipher*, CSPRNG *chaos*, fungsi linier.**

I. PENDAHULUAN

National Institute of Standards and Technology (NIST) menetapkan bahwa *substitution-box* (*s-box*) menjadi salah satu teknik dasar yang diperlukan dalam merancang blok *cipher*. Secara algoritme, *s-box* dapat menghasilkan hubungan nonlinier antara kunci dengan *ciphertext* dan memberikan efek *confusion* dari prinsip Shannon. Algoritme blok *cipher* yang ditetapkan sebagai standar pengamanan informasi dunia, seperti DES dan kemudian digantikan AES, keduanya, menggunakan *s-box* yang bersifat statis. Kondisi statis yang dimaksud adalah nilai pada setiap entri selalu tetap dan secara fungsi bersifat satu-ke-satu. Apabila hasil masukan-keluaran diekspresikan dalam grafik/diagram, maka pola dari *dataset* pada masukan tetap terlihat pada keluaran. Hal ini akan mempermudah *cryptanalyst* untuk melihat pola dan kemudian dapat memprediksi perilaku dari pola berdasarkan masukan, sehingga kompleksitas waktu dan ruang untuk melakukan *cryptanalysis* menjadi lebih pendek.

Kelemahan *s-box* statis dapat diperbaiki dengan membuatnya menjadi fleksibel, sehingga keluaran tidak menggambarkan pola dari masukan. Makalah ini mendesain *s-box* fleksibel menggunakan fungsi linier sebagai media transformasi, dan ditempatkan pada setiap entri *s-box*, dengan koefisien dan konstanta diperoleh dari pembangkitan bilangan acak dengan skema CSPRNG *chaos*. Setiap pembangkitan bilangan acak memerlukan kunci sebagai inisialisasi, sehingga perubahan pada kunci akan mengakibatkan perubahan pada *s-box* juga.

Pengertian fleksibel dalam kondisi substitusi dapat mengakomodasi relasi yang bersifat satu-ke-banyak, sehingga akan meningkatkan efek *diffusion* dan tentunya akan membuat para *cryptanalyst* sulit untuk melihat hubungan masukan-keluaran. Penggunaan bilangan acak pada koefisien dan konstanta dilakukan untuk menghindari fungsi yang sama pada masing-masing entri, sehingga pada akhirnya bila masukan sama, akan dihasilkan nilai yang berbeda dalam sebuah blok pesan. Fungsi linier digunakan pada setiap entri *s-box*, karena secara otomatis akan mempunyai inversi, sehingga tidak akan mengalami masalah dalam membuat inversi *s-box* yang biasanya digunakan dalam proses dekripsi. Fungsi yang berbeda pada setiap entri akan memperbesar ruang kunci dalam penebakan secara langsung dan akan mempersulit *cryptanalyst* melakukan *cryptanalysis*.

II. *S-BOX* FLEKSIBEL

Makalah ini melihat keterkaitan dengan penelitian yang telah dilakukan sebelumnya yang dapat digunakan sebagai acuan dan atau pembanding. Referensi [1] memandang *s-box* AES memiliki kelemahan dan melakukan modifikasi dengan menambahkan berbagai operasi ke dalamnya. Asumsi yang digunakan dalam penelitian tersebut adalah semakin banyak dan rumit algoritme akan membuat kriptografi akan lebih baik. Penelitian lain juga menggunakan asumsi yang sama [2], [3].

[1,2] *Dosen, Universitas Kristen Satya Wacana, Jl. Diponegoro 52-60, Salatiga. 50711.* (*e-mail: bambang.susanto@uksw.edu*) *alzdanny.wowor@uksw.edu*)

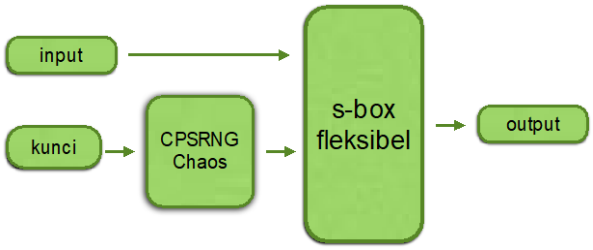
[3] *Mahasiswa, Master in Cyber Security, Radboud University, Houtlaan 4, 6525 XZ, Nijmegen, Netherland* (*vania.liwandouw@student.ru.nl*)

Sedangkan [4] memandang bahwa kecepatan komputasi saat ini perlu diimbangi dengan memuat algoritme yang berlapis karena rasio pembanding tidak imbang antara keamanan dan waktu proses. Penelitian yang dilakukan melihat dengan perspektif yang berbeda, yaitu kecepatan komputasi berdasarkan perangkat keras tidak diperhitungkan. Perspektif yang digunakan adalah kesederhanaan algoritme tetapi tetap memperhatikan tingkat keamanan.

Makalah ini berfokus pada cara menghasilkan s-*box* yang fleksibel, yaitu *s-box* akan berubah berdasarkan kunci. *S-box* dibangun menggunakan fungsi linier seperti pada [5] dan pembangkitan bilangan acak berbasis *chaos* menggunakan fungsi logistik seperti pada [6]. Kedua penelitian tersebut dijadikan sebagai acuan dalam mendesain sebuah algoritme blok *cipher*.

III. RANCANGAN ALGORITME

Konsep fleksibel pada algoritme adalah suatu proses yang mengubah nilai tetapan *s-box* berdasarkan kunci, sehingga perubahan pada kunci secara otomatis mengubah *s-box* juga. Skema global dari rancangan algoritme *s-box* fleksibel diberikan pada Gbr. 1.



Gbr. 1 Rancangan algoritme.

Kunci yang biasanya digunakan untuk proses enkripsi-dekripsi diolah untuk menjadi inisialisasi pada fungsi logistik (sebagai generator) untuk menghasilkan bilangan acak dalam skema CSPRNG *chaos*. Setiap bilangan acak digunakan sebagai konstanta dan koefisien dari fungsi linier yang menempati setiap entri *s-box*. Selanjutnya, dibahas bagian-bagian dari perancangan *s-box* fleksibel.

A. Pembangkitan CSPRNG Chaos

Bilangan acak yang dibangkitkan dengan skema CSPRNG *chaos* menggunakan fungsi logistik $f(x) = x(1 - x)$ [7], yang dalam bentuk iterasi diberikan pada (1).

$$x_{i+1} = x_i(1 - x_i) \quad (1)$$

Setiap nilai r dan x_0 adalah inisialisasi yang diperlukan untuk memperoleh bilangan yang bersifat *chaos* atau tidak. Berdasarkan [7], nilai inisialisasi untuk x_0 adalah bilangan desimal dan bilangan r adalah bilangan bulat.

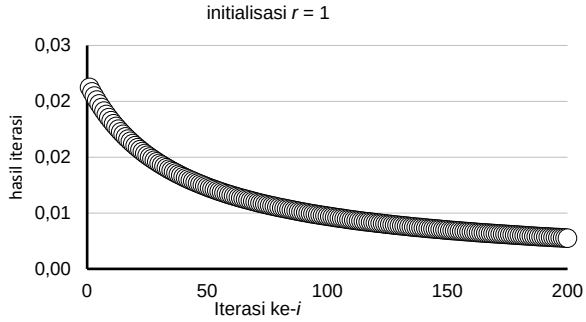
Proses untuk memperoleh nilai x_0 dilakukan dengan mengonversi setiap karakter kunci ke basis desimal yang dikalikan dengan nilai indeks kemudian dibagi dengan rata-rata kunci. Sebagai contoh, pada Tabel I, diambil kunci "jam 9 pm".

Nilai x_0 adalah perbandingan dari perkalian nilai indeks dengan angka desimal digunakan sebagai absis. Nilai ordinat adalah nilai rata-rata bilangan ASCII dari kunci, sehingga nilai absis diperoleh (1 × 106) + (2 × 97) + . . . + (8 × 109) dan nilai ordinat adalah (106 + 97 + 109 + 32 + 57 + 32)/6, sehingga diperoleh $x_0 = 0{,}024988457987$.

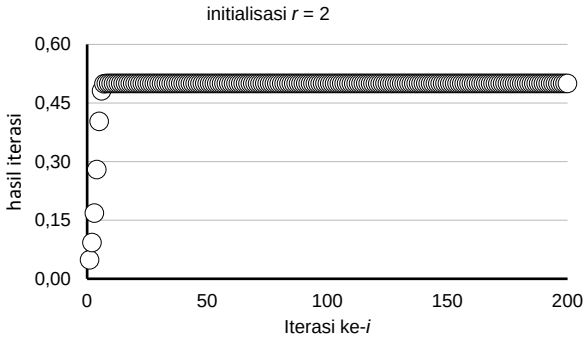
TABEL I
PENENTUAN NILAI x_0

Indeks	1	2	3	4	5	6	7	8
Kunci	j	a	m		9		p	m
ASCII	106	97	109	32	57	32	112	109

Sedangkan untuk nilai r dipilih bilangan bulat. Sebagai percobaan awal, dipilih r = 1, 2, 3, dan 4, dengan pengambilan 200 nilai awal.



(a)



(b)

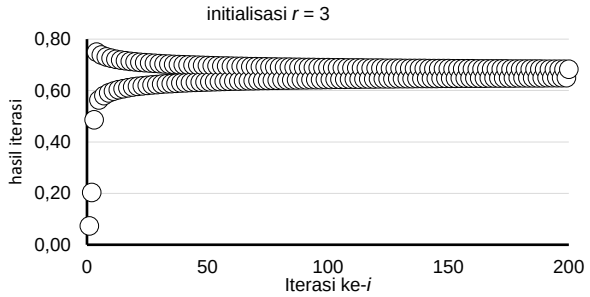
Gbr. 2 *Scatter plot* dengan $x_0 = 0{,}024988457987$; (a) r = 1; (b) r = 2.

Hasil untuk inisialisasi r = 1 diberikan pada Gbr. 2(a). Secara visual tampak belum membentuk *chaos*, tetapi membentuk pola berupa sebuah fungsi logaritma, dan konvergen pada sebuah titik. Sedangkan dengan mengambil nilai r = 2, hanya sembilan nilai di awal yang berbeda, tetapi setelah nilai ke-10 dan seterusnya membentuk garis linier kembali di titik 0,5. Kedua kondisi yang membentuk pola tentunya akan dihindari untuk dijadikan sebuah kunci karena belum membentuk sebuah *chaos*.

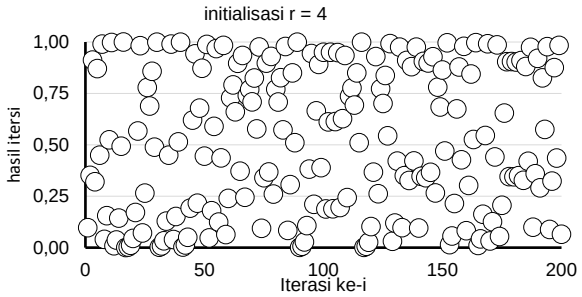
Inisialisasi r = 3 ditunjukkan pada Gbr. 3. Terlihat fungsi membentuk *biffurcation* atau pecah menjadi dua bagian, tetapi kembali lagi, masih membentuk sebuah pola. Ketika diambil r = 4, nilai *chaos* sudah diperoleh. Secara keseluruhan, untuk perlakuan nilai r yang menyatakan sifat nonlinier dengan *a* faktor nonlinier juga ikut meningkat untuk $0 < r < 4$.

Iterasi 200 nilai pertama untuk r = 4 adalah bilangan acak dengan *chaos* yang dihasilkan dalam bentuk desimal atau berupa bilangan riil antara 0 dan 1, padahal dalam kriptografi harus berupa bilangan bulat. Oleh karena itu, untuk

mendapatkan bilangan dilakukan dengan memilih urutan bilangan pada mantisa dari setiap hasil iterasi. Misalnya dipilih tiga digit untuk mengakomodasi banyak bilangan ASCII, dengan mengambil urutan ke-3, ke-4, dan ke-5 dari hasil iterasi 0,097456139818000, sehingga diperoleh 745.



(a)



(b)

Gbr. 3 *Scatter plot* dengan x_0 = 0,024988457987; (a) r = 3; (b) r = 4.

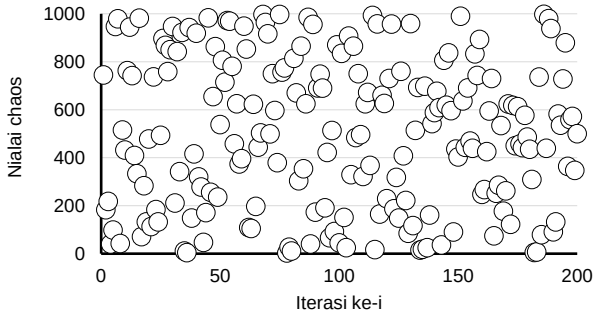
TABEL II
PEMBANGKITAN BILANGAN ACAK

i	x_i	***bil*-1**	***bil*-2**	***bil*-3**
1	0,097456139818000	745	613	981
2	0,351833762519114	183	376	251
3	0,91218706428303	218	706	428
4	0,320407296150948	040	729	615
5	0,870985842896747	098	584	289
6	0,449478017480762	947	801	748
7	0,989790117129303	979	011	712
8	0,0404225646498534	042	256	464
9	0,155154323667927	515	432	366
10	0,524325838060301	432	583	806

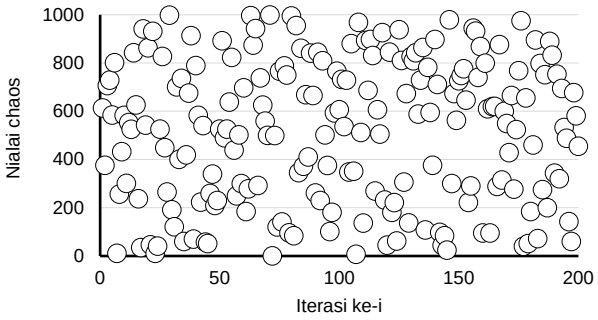
Tabel II adalah hasil sepuluh iterasi pertama (x_i) dari Gbr. 3 untuk r = 4. Pengambilan bilangan bulat pada mantisa dilakukan untuk mendapatkan tiga *dataset*, yaitu *bil*-1 untuk urutan ke-3, ke-4, dan ke-5, *bil*-2 untuk urutan ke-6, ke-7, dan ke-8, kemudian *bil*-3 untuk urutan ke-9, ke-10, dan ke-11 dari mantisa.

Kondisi *chaos* dapat dilihat berdasarkan sebaran bilangan menggunakan *scatter plot* pada koordinat Cartesian. Pengujian untuk melihat peluang nilai yang sama dapat didekati dengan pencarian nilai modus dari *dataset* yang ada. Sebanyak 200 iterasi bilangan pertama *bil*-1 ditunjukkan pada Gbr. 4.

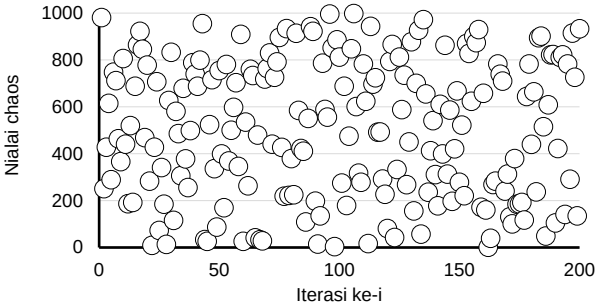
Kondisi *chaos* terjadi dengan persebaran nilai iterasi pada interval $0 \leq x_i \leq 1.000$. *Dataset* dari *bil*-1 tidak mempunyai nilai modus. Hasil ini menunjukkan bahwa 200 bilangan pertama yang dibangkitkan adalah kumpulan bilangan yang sangat unik dan tidak berulang.



Gbr. 4 *Scatter plot* untuk *bil*-1.



Gbr. 5 *Scatter plot* untuk *bil*-2.



Gbr. 6 *Scatter plot* untuk *bil*-3.

Pengujian grafik dan nilai modus juga dilakukan pada *bil*-2 dan *bil*-3, yang secara berturut-turut diberikan pada Gbr. 5 dan Gbr. 6. Ruang hasil juga masih sama pada selang interval $0 \leq x_i \leq 1.000$ dan nilai modus juga tidak ditemukan pada *bil*-2 dan *bil*-3.

Pembangkitan bilangan acak berdasarkan fungsi logistik menghasilkan lima belas urutan bilangan pada mantisa yang memungkinkan untuk diambil sebagai bilangan bulat. Apabila pengambilan tiga digit, maka dapat diperoleh lima *dataset* yang berbeda.

B. Rancangan S-Box Fleksibel

Rancangan awal diberikan pada Gbr. 1, yaitu *s-box* menerima masukan dari bilangan acak yang dihasilkan oleh CSPRNG *chaos*. Setiap entri *s-box* ditempati satu buah fungsi linier yang membutuhkan nilai konstanta (a_{ij}) dan koefisien (b_{ij}). Secara umum dapat ditulis sebagai (2).

$$f_{ij}(x) = a_{ij}x + b_{ij} \quad (2)$$

dengan $i = 1, 2,\cdots, m$ dan $j = 1, 2,\cdots, n$.

$f_{11}(x)$	$f_{12}(x)$	...	$f_{1n}(x)$
$f_{21}(x)$	$f_{22}(x)$	...	$f_{2n}(x)$
⋮	⋮	⋱	⋮
$f_{m1}(x)$	$f_{m2}(x)$	...	$f_{mn}(x)$

Gbr. 7 Rancangan *s-box* fleksibel.

$f_{11}(x) = 745x + 613$	$f_{12}(x) = 183x + 376$	$f_{13}(x) = 706x + 218$	$f_{11}(x) = 40x + 729$
$f_{21}(x) = 98\ x + 584$	$f_{22}(x) = 947x + 801$	$f_{23}(x) = 979x + 11$	$f_{24}(x) = 42x + 256$
$f_{31}(x) = 515x + 432$	$f_{32}(x) = 763x + 301$	$f_{23}(x) = 944x + 553$	$f_{24}(x) = 742x + 525$
$f_{41}(x) = 409x + 842$	$f_{42}(x) = 333x + 626$	$f_{43}(x) = 982x + 237$	$f_{44}(x) = 71x + 35$

Gbr. 8 Rancangan *s-box* fleksibel dengan kunci "jam 9 pm".

$f_{11}(x) = 719x + 380$	$f_{12}(x) = 98x + 867$	$f_{13}(x) = 118x + 250$	$f_{14}(x) = 371x + 580$
$f_{21}(x) = 569x + 552$	$f_{22}(x) = 541x + 148$	$f_{23}(x) = 331x + 329$	$f_{24}(x) = 563x + 304$
$f_{31}(x) = 530x + 139$	$f_{32}(x) = 51x + 447$	$f_{23}(x) = 861x + 115$	$f_{24}(x) = 139x + 219$
$f_{41}(x) = 125x + 336$	$f_{42}(x) = 718x + 511$	$f_{43}(x) = 959x + 502$	$f_{44}(x) = 954x + 292$

Gbr. 9 Rancangan *s-box* fleksibel dengan kunci "jam 9 pn".

Penyusunan masing-masing fungsi dalam *s-box* ditunjukkan pada Gbr. 7. Penentuan banyak *m* dan *n* dapat disesuaikan dengan ukuran blok pada kriptografi. Misalnya kriptografi dalam ukuran 128 bit atau ekuivalen dengan 16 *byte*, sehingga diperoleh 16 bilangan desimal, maka *s-box* dapat dibuat dengan $m = n = 4$, atau dapat juga digunakan $m = 8$ dan $n = 2$ atau sebaliknya, $m = 2$ dan $n = 8$.Fungsi linier pada setiap entri *s-box* dihasilkan ketika kunci dimasukkan oleh pengguna. Maka, otomatisasi penentuan nilai koefisien dan konstanta dapat dilakukan dengan regenerasi dari CSPRNG *chaos*. Pada Tabel II, *bil*-1 dan *bil*-2 dapat digunakan sebagai konstanta ataupun koefisien. Persamaan (2) merupakan fungsi linier dengan banyak baris *i* dan *j* sebagai kolom. Sebuah entri mewakili sebuah karakter, sehingga apabila rancangan kriptografi dalam 64 bit, maka dipilih 16 kotak dengan $i = 4$, $j = 4$, sehingga akan membutuhkan 32 bilangan acak sebagai konstanta dan koefisien. *S-box* fleksibel diperoleh dengan menggunakan *bil*-1 dan *bil*- 2 sebagai konstanta dan koefisien, seperti ditunjukkan pada Gbr. 8.

Setiap kunci yang dimasukkan akan memberikan nilai inisialisasi pada fungsi pembangkit yang berbeda, sehingga fungsi linier pada *s-box* secara otomatis ikut berubah. Gbr. 8 adalah *s-box* yang dihasilkan dari kunci "jam 9 pm". Apabila kunci diubah sebuah bit sehingga menjadi "jam 9 pn" (dengan $m = 01101101_2 = 109_{10}$, dan $n = 01101110_2 = 110_{10}$), dihasilkan *s-box* dengan konstanta dan koefisien berbeda secara sangat signifikan, seperti ditunjukkan pada Gbr. 9.

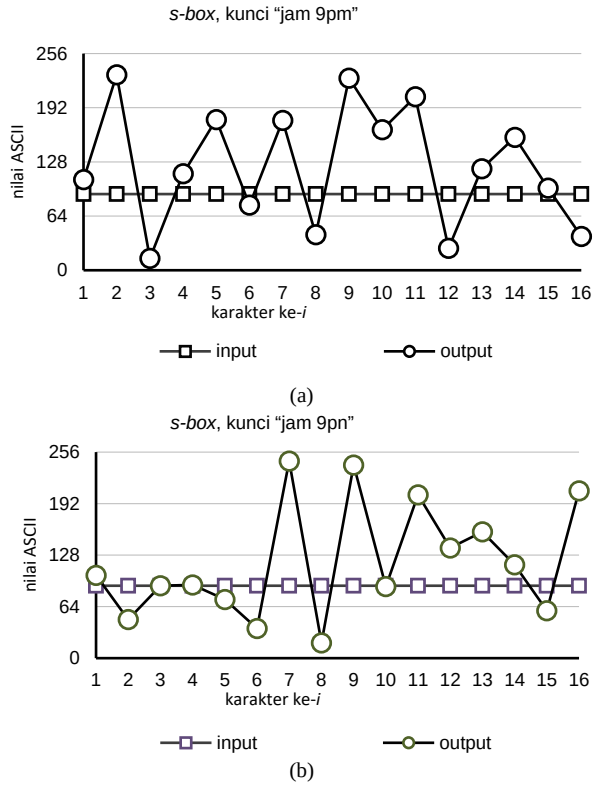
Fleksibelnya *s-box* ditentukan dari perubahan kunci yang dimasukkan, sehingga menghasilkan nonliniernya sebuah relasi masukan-keluaran. Perubahan kecil pada masukan kunci mengakibatkan perubahan yang besar terjadi pada *s-box*. Dengan perubahan 1 bit ini dapat dikatakan bahwa bilangan acak CSPRNG *chaos* mempunyai sifat *butterfly effect*, sehingga bilangan acak yang dihasilkan memenuhi sifat *diffusion* dari prinsip Shannon.

Kekuatan dari pembangkitan bilangan acak tentu akan memberikan pengaruh yang besar dari sisi kekuatan algoritme, karena konstanta dan koefisien secara otomatis ikut berubah.

IV. PENGUJIAN ALGORITME

A. *Pengujian S-Box Fleksibel*

Pengujian dilakukan untuk melihat kekuatan dari *s-box* fleksibel. Berdasarkan [8], digunakan dua pengujian, yaitu uji masukan yang sama (dengan karakter Z) dan masukan yang bervariasi. Kekuatan dari *s-box* fleksibel terjadi karena perubahan pada kunci. Oleh karena itu, diuji perubahan 1 bit dan dilihat seberapa besar perubahan yang terjadi pada keluaran *s-box*. Hasil dari pengujian berbeda kunci berdasarkan masukan ditunjukkan pada Gbr. 10 dan Gbr. 11.

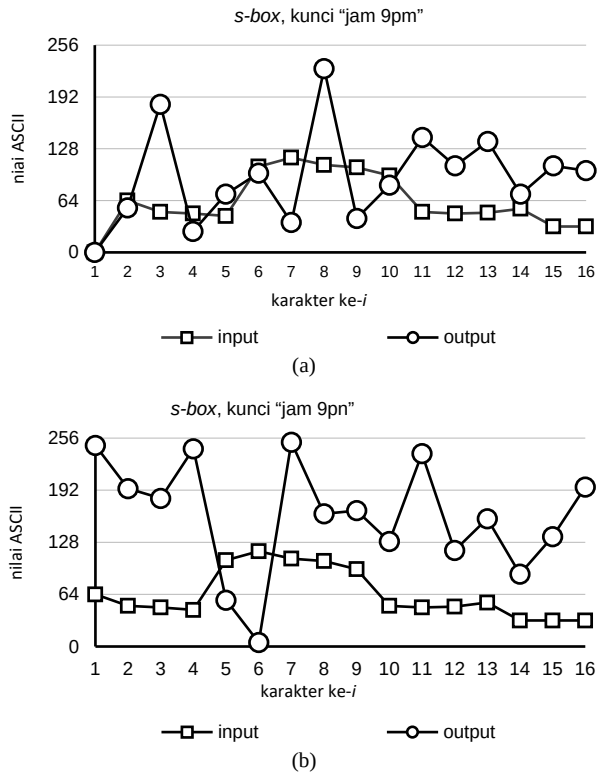


Gbr. 10 Pengujian *s-box* fleksibel dengan masukan "ZZZZZZZZZZZZZZZZ"

Perubahan sebuah bit pada kunci menunjukkan keluaran yang tidak sama dan tidak ada pola yang memudahkan untuk proses penebakan secara langsung, sehingga algoritme *s-box* fleksibel mempunyai sifat *butterfly effect*.

Pengujian kedua dilakukan dengan masukan bervariasi "@20-juli 2016" sebagai kombinasi huruf, simbol, dan angka untuk melihat kekuatan algoritme terhadap variasi masukan.

Grafik yang diperoleh pada Gbr. 11 memberikan informasi bahwa kedua *s-box* sangat baik dalam melakukan proses substitusi. Hal ini terlihat pada relasi antara masukan dan keluaran tidak dalam relasi satu-ke-satu.



Gbr. 11 Pengujian *s-box* fleksibel dengan masukan "@20- juli 2016".

Kedua *s-box* juga dapat memberikan hasil yang sangat berbeda walaupun pada kunci hanya diubah sebuah bit, sehingga dengan masukan yang bervariasi nilai keluaran yang dihasilkan juga tetap bervariasi.

Hasil yang diperoleh pada Gbr 11 menunjukkan pola masukan tidak tergambarkan pada pola keluaran. Berdasarkan [9], *s-box* fleksibel memiliki karakteristik *diffusion* karena statistik kunci terserap atau menghilang pada statistik keluaran. Kekuatan *diffusion* pada *s-box* melanjutkan kekuatan *diffusion* yang sebelumnya telah diperoleh pada pembangkitan kunci, sehingga membuat algoritme lebih baik dalam menahan serangan dari *cryptanalysis* dalam melihat hubungan antara masukan dan keluaran.

TABEL III
UJI KORELASI PADA *S-BOX* FLEKSIBEL

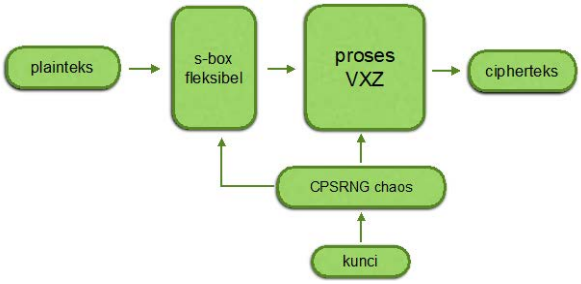
Masukan	***s-box* fleksibel "jam 9 pm"**	***s-box* fleksibel "jam 9 pn"**
Dipo 52-66	0,005	0,110
fti uksw	-0,184	-0,265
@20-juli 2016	0,056	-0,274
ZZZZZZZZ	0,127	0,140
s4lat1g4	0,083	-0,073

Pengujian secara statistik juga dilakukan menggunakan nilai korelasi, sehingga dapat dilihat seberapa kuat hubungan antara variabel masukan (X) dan keluaran (Y). Nilai korelasi yang diperoleh dalam kriptografi diharapkan mendekati nol, sehingga dapat dilihat kekuatan algortime dalam menyamarkan sebuah informasi. Hasil uji korelasi pada *s-box* fleksibel disajikan dalam Tabel III

Dari pengujian nilai korelasi secara rata-rata diperoleh 0,091 pada *s-box* "jam 9 pm" dan 0,172 untuk *s-box* "jam 9 pn". Oleh karena itu, *s-box* fleksibel yang dirancang dapat menghilangkan hubungan secara statistik antara masukan dan keluaran. Konsistensi *s-box* dalam mengubah dua nilai masukan menunjukkan kekuatan algoritme dari sebuah proses substitusi.

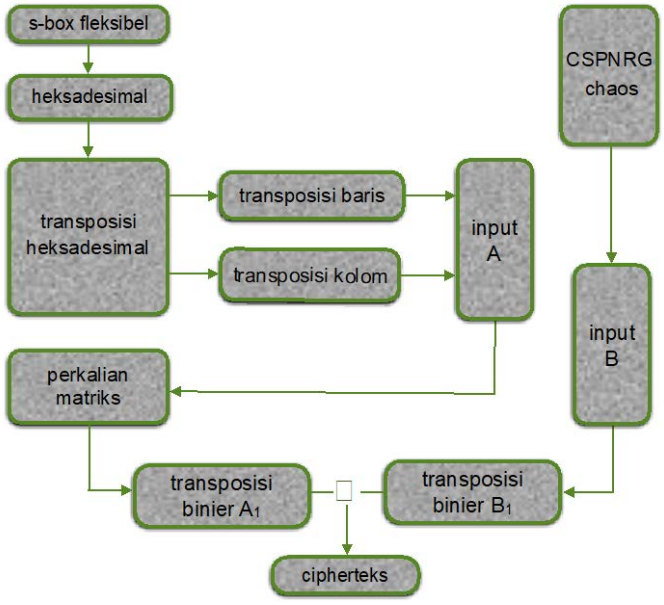
B. Pengujian S-Box Fleksibel pada Blok Cipher

S-box fleksibel diuji dengan merancang sebuah blok *cipher* dengan skema super enkripsi, dan transposisi sederhana dilakukan secara baris dan kolom. Rancangan secara umum diberikan pada Gbr. 12.



Gbr. 12 Rancangan global skema blok *cipher*.

Rancangan blok *cipher* fokus pada pembangkitan kunci untuk menghasilkan bilangan acak berbasis CSPRNG *chaos*. *S-box* fleksibel juga digunakan pada proses VXZ yang menggabungkan *plaintext* dan kunci, yang secara jelas diberikan pada Gbr. 13.



Gbr. 13 Proses VXZ dalam blok *cipher*.

Algoritme pada proses VXZ tetap menggunakan beberapa skema dari blok *cipher* lain, seperti transposisi yang dilakukan secara kolom maupun baris dalam basis heksadesimal maupun

biner. Perkalian matriks juga digunakan untuk memperoleh sifat satu-ke-banyak (*one-to-many*) antara masukan *A* dengan tetapan bilangan konstanta. Berdasarkan Gbr 13, *ciphertext* diperoleh dengan melakukan XOR antara transposisi biner A_1 dan transposisi biner B_1.

Simulasi enkripsi dilakukan dengan mengambil *plaintext* "FTI UKSW", kunci "jam 9 pm", dan *s-box* fleksibel pada Gbr. 8. Diperoleh hasil substitusi *plaintext* dalam modulus 256 seperti disajikan pada Gbr. 14.

	1	2	3	4
1	137	18	3	55
2	183	135	144	71
3	26	246	155	67
4	72	42	208	97

Gbr. 14 Keluaran *s-box* fleksibel dengan masukan "FTI UKSW".

Langkah pertama yang dilakukan pada proses VXZ adalah melakukan konversi hasil *s-box* ke dalam heksadesimal, kemudian proses transposisi heksadesimal dengan menggunakan aturan Ambil a, Geser b.

Proses ini mengambil pada posisi karakter ke-a (dengan $a = 1, 2, \cdots, 8$) dan kemudian pergeseran dilakukan sejauh b. Apabila pergeseran ke kiri, ditulis $-b$, sebaliknya untuk pergeseran ke kanan, ditulis $+b$.

Ambil x, Geser y	urutan bit ke-															
1, 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
89B71A481287F62A	8	9	B	7	1	A	4	8	1	2	8	7	F	6	2	A
Ambil x, Geser y	urutan bit ke-															
1, +2	2	4	6	8	10	12	14	16	1	3	5	7	9	11	13	15
90D04761039B3743	0	0	7	1	3	B	7	3	9	D	4	6	0	9	3	4

Gbr. 15 Proses transposisi Ambil *a*, Geser *b*.

Gbr. 15 menunjukkan proses transposisi, yaitu (1,0) menunjukkan ambil karakter ke-1 dan menggeser sejauh 0 karakter, sedangkan (1,+2) menunjukkan bahwa karakter awal yang diambil adalah posisi ke-1 dan pergeseran dilakukan ke kanan sejauh dua karakter.

Selanjutnya adalah setiap nilai dimasukkan dalam dua buah matriks berukuran 4×4 dan dilakukan transposisi secara kolom dan juga secara baris. Hasil dari masing-masing transposisi kemudian digabung sehinggga diperoleh 16 karakter ASCII yang menjadi masukan *A*. Sedangkan masukan *B* adalah hasil pembangkitan dari CSPRNG sebagai *bil*-3 (dalam mod 256) pada Tabel II.

Perkalian matriks dilakukan berdasarkan (IA × NT) × (IA), dengan NT = nilai tetapan dan IA = masukan *A*. Misalnya, pada entri $(IA \times NT)_{(i=1,j=1)}$ diperoleh 214 = (184·52) + (145·145) + (183·147) + (114· 160). Kemudian $((IA \times NT) \cdot IA)_{(i=1,\ j=1)}$ = 208 (mod 256). Hasil operasi perkalian secara lengkap ditunjukkan pada Gbr. 16.

Hasil perkalian matriks dikonversi ke biner dan dilakukan proses *transposisi biner* A_1 dan juga untuk proses *transposisi biner* B_1 yang memproses masukan *B*. Pada hasil kedua transposisi dilakukan proses XOR untuk mendapat *ciphertext*, seperti yang diperlihatkan pada Tabel IV.

Nilai Tetapan (NT)				Input A (IA)				IA x NT				(IA x NT) · IA			
52	131	7	170	184	145	183	114	214	46	23	61	208	14	113	42
145	61	62	96	68	137	22	141	43	117	14	92	108	157	52	172
147	35	11	31	115	48	52	9	8	63	171	140	152	208	188	236
160	138	218	82	162	159	186	118	133	211	106	134	42	13	4	196

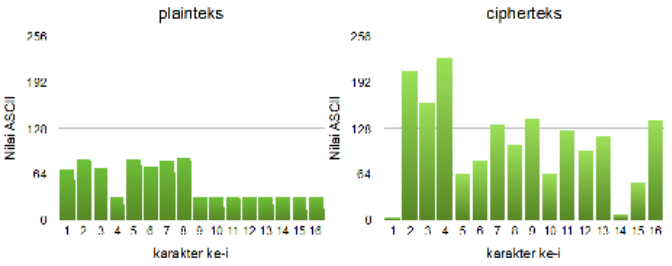
Gbr. 16 Perkalian matriks pada proses VXZ.

TABEL IV
CIPHERTEXT DARI RANCANGAN BLOK *CIPHER*

Chipertext Heksadesimal	03408D74D0534009A4867 E35E268608B
Chipertext Biner	00000011010000010001101 01110100110100000101001 10 01000000000100110100100 10000110011111000110101 11100010011010000110000 0 10001011

C. *Pengujian Blok Cipher 128 bit*

Pengujian *s-box* fleksibel sebagai sebuah proses substitusi pada blok *cipher* akan dilihat berdasarkan perbedaan histogram antara *plaintext* dan *ciphertext*. Sebagai sebuah kriptografi, sangat dimungkinkan untuk setiap *user* mengamankan pesan dengan variasi karakter. Oleh karena itu, pengujian juga dilakukan dengan memperhatikan variasi *plaintext* [8]. Pengujian yang pertama adalah *plaintext* "FTI UKSW". Pilihan ini digunakan karena *user* biasanya hanya mengirim pesan dalam karakter abjad saja.



Gbr. 17 Histogram untuk *plaintext* "FTI UKSW".

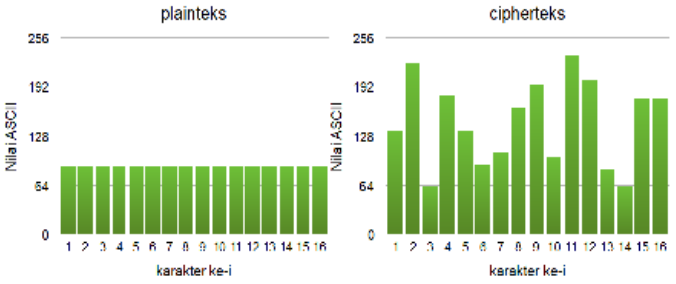
Histogram *plaintext* pada Gbr. 17 tampak berbeda hanya pada karakter ke-1 sampai ke-8, karena karakter ke-9 sampai ke-16 adalah *padding* dengan spasi yang ekuivalen dengan angka 32 dalam ASCII, sehingga tampak frekuensi yang sama. Bila dilihat berdasarkan nilai frekuensi, pola pada *plaintext* tidak tampak pada *ciphertext*, sehingga algoritme blok *cipher* yang dirancang dapat menyembunyikan atau terserap sehingga tidak tampak di *ciphertext*.

Pengujian yang lain adalah dengan menghitung nilai korelasi antara *plaintext* dan *ciphertext*. Walaupun secara teori, pada korelasi perlu dilihat hubungan kedua variabel berlangsung secara positif atau negatif, tetapi dalam pengujian kriptografi

faktor tersebut tidak diperhatikan. Data penting dari korelasi adalah nilainya mendekati 0 (baik secara negatif maupun positif) atau 1 (atau -1). Jika nilai mendekati 0, maka algoritme dapat dikatakan menyamarkan *plaintext*. Sebaliknya, apabila mendekati 1 atau -1, maka algoritme dapat dikatakan gagal dan perlu untuk dirancang kembali.

Nilai korelasi yang diperoleh *plaintext* "FTI UKSW" adalah −0,36. Berdasarkan [10], sebagai penentuan tingkat hubungan, nilai korelasi ini berada pada kategori "rendah" dan hal ini menunjukkan bahwa algoritme yang dirancang dapat membuat *plaintext* dan *ciphertext* tidak berhubungan secara statistik.

Pengujian yang kedua terhadap rancangan blok *cipher* menggunakan *plaintext* yang sama, yaitu Z sebanyak 16 karakter. Nilai masukan Z sudah digunakan oleh pengujian sebelumnya. Pengujian ini diperlukan karena secara bit 01011010 adalah simetris. Sifat simetris ini kadang menggagalkan proses transposisi, karena digunakan kotak 8 × 8, maka proses akan tidak memberikan hasil yang maksimal [8].



Gbr. 18 Histogram *plaintext* dan *ciphertext* dengan masukan "ZZZZZZZZZZZZZZZZ".

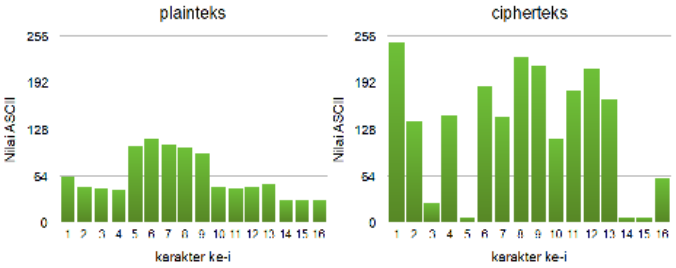
Hasil dari pengujian Z diperlihatkan pada Gbr. 18. Secara visual, algoritme dapat menghilangkan pola dari *plaintext*, sehingga menghilangkan hubungan satu-ke-satu yang menyulitkan *cryptanalysis* untuk melakukan penebakan secara langsung.

Nilai korelasi untuk pengujian Z tidak dapat diperoleh, karena untuk nilai yang sama, ordinat pada persamaan korelasi menghasilkan nilai 0, sehingga menjadi tidak terdefenisi. Tidak dapat dihitung secara korelasi bukan berarti algoritme yang dirancang tidak dapat melakukan proses dengan baik, bahkan sebaliknya, walaupun diberikan masukan yang sama, tetapi *ciphertext*-nya sangat bervariasi.

Pengujian terakhir adalah *plaintext* dari kombinasi karakter [8], yang terdiri atas abjad, angka, dan simbol, yaitu "@20-juli 2016". Secara prinsip, hampir sama dengan pengujian *plaintext* sebelumnya, hanya saja ruang domain yang diperbesar. Hasil pengujian diberikan pada Gbr. 19.

Hasil yang diamati adalah hubungan antara kedua variabel. Histogram yang dihasilkan juga tampak tidak ada relasi secara langsung. Diperoleh nilai korelasi adalah −0,22, sehingga berada pada kategori "rendah" juga.

Ketiga pengujian yang dilakukan menggunakan histogram dan uji korelasi menunjukkan bahwa rancangan algoritme blok *cipher* mempunyai karakter *diffusion* dari Shanonn, karena statistik pada *plaintext* terserap atau menghilang pada statistika *ciphertext* [9].



Gbr. 19 Histogram *plaintext* dan *ciphertext* dengan masukan "@20-juli 2016".

D. Pemenuhan Prinsip Cipher Berulang

Salah satu prinsip blok *cipher* adalah *cipher* berulang (*iterated cipher*), yaitu proses putaran yang memasukkan *ciphertext* kembali sebagai *plaintext*, dengan setiap putaran menggunakan kunci yang berbeda. Rancangan ini membangkitkan bilangan acak berdasarkan CSPRNG *chaos* sebanyak kebutuhan putaran. Sebagai contoh, dilakukan 15 putaran, sehingga dibangkitkan 15 × 16 × 3 = 720 bilangan acak secara iteratif.

TABEL V
NILAI KORELASI DARI SETIAP PUTARAN

Putaran	FTI UKSW	@20-Juli_2016
1	0,263	0,226587
2	0,001	0,245549
3	0,146	0,323023
4	0,086	0,070751
5	0,131	0,107954
6	0,172	0,070016
7	0,104	0,045489
8	0,214	0,038607
9	0,010	0,303827
10	0,158	0,139372
11	0,230	0,139372
12	0,175	0,075378
13	0,043	0,085551
14	0,140	0,285076
15	0,193	0,116099
Rata-rata	0,137	0,151

Proses *cipher* berulang dilakukan agar memperkuat karakteristik *diffusion* dan *confusion* dari teori Shannon. Makalah ini juga melihat seberapa baik proses perulangan dilakukan terhadap kekuatan algoritme. Tabel V menyajikan nilai korelasi yang dilakukan terhadap 15 putaran dengan mencoba dua *plaintext*. Kedua *plaintext* secara rata-rata mempunyai nilai 0,144. Hal ini menunjukkan bahwa setiap putaran yang dilakukan algortime yang dirancang dapat mempertahankan nilai korelasi yang selalu mendekati nol.

E. Pengujian Avalanche Effect

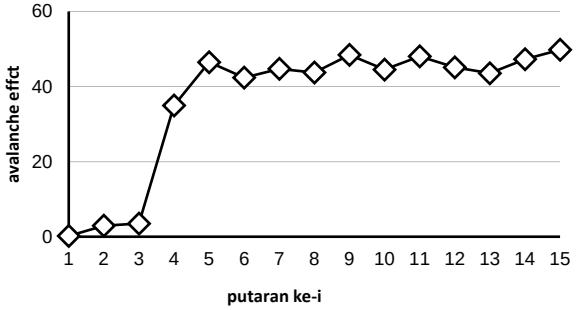
Pengujian *Avalanche Effect* (AE) dilakukan untuk melihat perubahan kecil pada *plaintext* memberikan pengaruh yang besar terhadap *ciphertext* atau tidak. Tabel VI adalah nilai AE berdasarkan putaran, dengan pengujian *plaintext* "DISASTER", kemudian diganti dengan "DISCSTER". Perubahan kedua *plaintext* hanya pada perbedaan 1 bit, yaitu dari A: 01100001

ke C: 01100011. Kunci yang digunakan adalah “SRIRAMSR”. Penggunaan *plaintext* dan kunci ini untuk dapat dibandingkan dengan penelitian sebelumnya terkait nilai AE pada [11].

TABEL VI
NILAI *AVALANCHE EFFECT* SETIAP PUTARAN

Putaran	Banyak Bit Berbeda	*Avalanche Effect*
1	1	0,2
2	15	2,93
3	18	3,52
4	179	34,96
5	238	46,48
6	217	45,12
7	229	46,61
8	224	49,45
9	248	49,63
10	228	47,53
11	246	48,05
12	231	49,75
13	223	49,62
14	242	47,27
15	255	49,60
Rata-rata		38,04

Nilai AE tampak meningkat sejak putaran pertama, tetapi setelah putaran ke-5 tampak nilai AE sudah berosilasi dan mempunyai kecenderungan yang tidak meningkat lagi. Pengujian putaran selanjutnya tidak dilakukan lagi, karena berdasarkan grafik Cartesian yang diberikan pada Gbr. 20, tampak sudah mengalami kejenuhan.



Gbr. 20 Histogram *plaintext* “FTI UKSW”.

Tingkat kejenuhan dapat diperhatikan dengan melihat grafik yang sudah tidak mengalami kenaikan atau penurunan secara signifikan, tetapi mengalami fluktuasi secara terus menerus. Fluktuasi data memberikan informasi penting bagi penentuan banyak putaran yang dilakukan untuk sebuah algoritme dalam melakukan proses enkripsi-dekripsi. Berdasarkan data AE yang ada pada Tabel VI, banyak putaran yang direkomendasikan agar pengamanan lebih terjamin adalah > 5 putaran.

Banyak putaran akan membuat kebutuhan ruang dan waktu meningkat, tetapi bila terlalu sedikit akan mengurangi eksistensi dari kriptografi itu sendiri dalam mengamankan sebuah data. Dipilih 12 putaran untuk dapat mengetahui kekuatan dari algoritme yang dirancang, kemudian dibandingkan dengan algoritme kriptografi lain berdasarkan [11]. Hasil perbandingan ditunjukkan pada Tabel VII.

TABEL VII
PERBANDINGAN NILAI *AVALANCHE EFFECT*

Algoritme	Nilai *Avalanche Effect*
Caesar	1,56
Playfair Cipher	6,25
Vigenerre	3,13
DES	54,68
Ramanujan & Karupiah	70,31
Blowfish	28,71
Rancangan Algoritme	49,75

Rancangan algoritme berada pada urutan ketiga di bawah algoritme Ramanujan & Karuppiah dan DES. Kedua algoritme yang unggul menggunakan semua prinsip blok *cipher*, sedangkan algoritme yang dirancang tidak menggunakan jaringan Feistel. Walaupun demikian, algoritme yang dirancang mampu mengungguli algoritme *Blowfish*, yang sudah banyak digunakan sebagai pengamanan informasi.

V. KESIMPULAN

Pembangkitan bilangan CSPRNG *chaos* dapat memberikan bilangan acak yang berbeda sebagai kunci dan sangat sensitif terhadap perubahan nilai masukan kunci awal. Bahkan dengan perubahan 1 bit saja akan menghasilkan bilangan acak yang berbeda. Rancangan *s-box* fleksibel juga dapat memberikan kekuatan substitusi dengan baik, terlihat pada pengujian berdasarkan nilai korelasi antara masukan dan keluaran berhasil membuat kedua variabel uji tidak berhubungan secara statistika.

Kekuatan fleksibel dari *s-box* terlihat ketika perubahan masukan kunci, yang secara otomais akan menghasilkan *a* yang berbeda. Fleksibelnya *s-box* masih berlanjut ketika dilakukan proses putaran dengan masukan yang sama, *s-box* akan tetap berbeda untuk setiap putarannya. Hal ini tidak terdapat pada kriptografi lainnya yang menggunakan *s-box* statis. Kemudian, rancangan kriptografi blok *cipher* dapat memberikan kekuatan *confusion-diffusion* dari prinsip Shannon, karena perubahan yang kecil pada nilai *plaintext* maupun kunci memberikan perubahan yang sangat besar pada *ciphertext*.

Selain itu, prinsip blok cipher lain yang dipenuhi adalah *iterated cipher* dengan $n-$ putaran disarankan $n > 5$ dan banyak kebutuhan 46 n bilangan acak. Selanjutnya, dari tpngujian *avalanche effect* dengan mengambil $n = 15$ putaran dan menggunakan 690 bilangan acak diperoleh rancangan kriptografi memberikan hasil lebih baik dari algoritme kriptografi *Twofish* walaupun masih di bawah DES.

UCAPAN TERIMA KASIH

Ucapan terima kasih disampaikan kepada Badan Pusat Penelitian dan Pengabdian Masyarakat (BP3M) Universitas Kristen Satya Wacana atas dukungan dana penelitian melalui skema Penelitian Fundamental Internal pada tahun anggaran 2015.

REFERENSI

[1] J. Cui, L. Huang, H. Zhong, C. Chang, dan W. Yang, “An Improved AES S-Box and Its Performance Analysis,” *International Journal of Innovative Computing, Information and Control*, Vol.7, No. 5(A), hal. 2291-2302, 2011.

[2] K. Prasadh, K. Ramar, dan R. Gnanajeyaraman, "Public Key Cryptosystems Based on Chaotic Chebyshev Polynomials," *Journal of Engineering and Technology Research*, Vol. 1, No. 7, hal. 122-128, 2009.

[3] A. Ramadhanus adn F. Firdaus, "Blackfish: Block Cipher dengan Key-Dependent S-Box dan P-Box," Program Studi Teknik Informatika, STEI ITB, Laporan Akhir, Bandung, 2013.

[4] P. Irfan dan Y. Prayudi, "Penggabungan Algoritma Chaos dan Rivers Shamir Adleman (RSA) untuk Peningkatan Keamanan Citra," *Seminar Nasional Aplikasi Teknologi Informasi*, 2015, hal. D5– D10.

[5] V.B. Liwandouw dan A.D. Wowor, "Kombinasi Algoritma Rubik, CSPRNG Chaos dan S-Box Fungsi Linier dalam Perancangan Kriptografi Block Cipher," *Seminar Nasional Sistem Informasi Indonesia*, 2015, hal. 207–214.

[6] E.Y..I. Kurniawan, "Penerapan Teori Chaos pada Kriptografi Menggunakan Algoritma Stream Cipher dan Electronic Code Book (ECB) untuk Keamanan Pesan Teks," Skripsi, Universitas Dian Nuswantoro, Semarang, Indonesia, 2014.

[7] R. Munir, *Kriptografi*. Bandung, Indonesia: Informatika, 2006.

[8] V.B. Liwandouw dan A.D. Wowor, "Desain Algoritma Berbasis Kubus Rubik dalam Perancangan Kriptografi Simetris," *Seminar Teknik Informatika & Sistem Informasi*, 2015, hal. 42–47.

[9] R. Sadikin, *Kriptografi untuk Keamanan Jaringan*, Yogyakarta Indonesia: Penerbit Andi, 2012.

[10] D.C. Montgomerry dan G.C. Runger, *Aplied Statistics ad Probabality for Engineers*, New Jersey, USA: John Wiley & Sons, 2014.

[11] S. Ramanujam dan M. Karuppiah, "Designing an Algorithm with High Avalanche Effect," *International Journal of Computer Science and Network Security*, Vol. 11, No. 1, hal. 106-111, Jan. 2011.